

Code_</g/re/p>

Work In Progress Document

Member Roll number	Member Name
2018007	Aditya Singh Rathore
2018069	Pranav Jaggarwal
2018100	Shikhar Tiwari
2018320	Tanupriya Das
2018357	Pragya Gandhi

Week 1

10th January [Meeting 1]

After a lot of discussion on selection of project, following points were emphasized :

- This thing is 40% of total marking.
- There are 10% of Bonus marks. Innovation will play a huge role here.
- As we are devoting lot of time here, this can serve as a good project that we have done.

We all pitched in some ideas and finally narrowed down the options to the following :

- A shoe store that has multiple sellers.
- A Banking System.

Next meeting is planned for 12th January. Till then, we will all look for examples of these types of systems.

11th January [Meeting 2]

We all realized that we are basically reinventing the wheel. There was nothing innovative in implementing the a shoe store or a Banking System. However, while exploring for other ideas, we saw Github. We realised it would be a nice idea to work with the codes like github but also have additional features which Github lacked.

We decided to make a databases for programmers to provide them with the details of the code, provide a platform for them to share it and to ask their queries. The interesting part would be to create a database of functions, classes, interfaces, Packages and Libraries. Here is what we planned :

- User can create a repository. They can add **description** and **tags** to this repository. For eg. `Sorting-algorithms`.
- We chose the syntax of Java as our base for parsing the code.
- When the user creates a library or a package, he/she can add **description** to it along with **tags**.
- When the user writes code of a class in a file, they are required to add comments in a specific style, similar to `javadoc`.
- The information can be extracted by parsing the code. User can add **tags** and description about classes. For a function, he/she can explain `return` type and `parameters`.
- All this information will be stored in database.
- Now, if somebody searches `sorting`. We will display relevant classes/functions/packages/library etc.

We also identified the following additional features :

- A **Bug tracker** system where somebody can submit their bugs and ask for help. Others can follow a bug and if it feels good, follow it.
- A **Q&A** system where you can ask questions and find answers to them.
- User can also put price on his/her code and sell it.
- A user will have **reputations** which dictate behavior on the system.

We also identified the following stakeholders :

- **Programmers** : They can ask questions, share codes, solve bugs, ask for bugs, get their codes documented etc. As they are the main users, they will act as moderators, editors in the forum.
- **Communities** : Open source communities, educators etc., can share their codes in a much better way. They can also get contributions. They can share their knowledge through a **Q&A** portal.
- **Companies**: Companies can buy and sell codes. We have options for keeping a repository `public` or `private`. Companies can document their codes there.
- **Recruiters**: Recruiters can see how many bugs a user has solved, his reputation, his answers and his repositories and then can directly contact them, if they want.

Week 2

15th January [Meeting 3]

Here are the questions our system will answer to stakeholders:

- **Programmers**: Our system is for programmers, by programmers, of the programmers.
 - Programmers can document their codes in a convenient manner.
 - They can search for an algorithm or data structure and see its implementation
 - Ask questions in community forums
 - Solve and ask for solutions to bugs
 - Sell their codes and earn money
 - Gain employment
 - Become moderators and editors in the forum.
- **Communities**: By communities, we identified open source communities and educators

- The communities can share their code. As we have a tag system, their code will be searchable.
- They can ask and answer their own question, sharing their knowledge.
- Find code from other communities.
- Ask help in solving a bug.
- Reputed communities can act as moderators and editors in Q&A website.
- **Companies:** Companies can buy codes and sell codes. They can also get a very good documentation of their code.
 - As the code is organized very well, a programmer of company won't spend time looking through code to what it does.
 - Company can buy a piece of code from a free lance coder.
 - Company can offer prizes on solving bugs in their code.
 - Companies can write pieces of software that others may be interested in and earn money.
 - Start a community driven project.
- **Recruiters :** The recruiter is here to find good programmers. Our system will answer their queries as we can **profile user data**. We can identify a good programmer by number of bugs solved etc. Here are questions that our system can answer for recruiter which will help him/her find a good programmer.
 - To find a programmer with reputation > x.
 - To find someone who has implemented a software, for eg. game engine or worked on it.
 - To find someone who has solved at least y bugs with tag says, "POSIX Threads".
 - To find the user who answered a particular question which had significant number of upvotes.
 - To find someone who has sold some codes or their repositories that are shared by many.

Following are the data entities that we have identified:

- Repository
- File
- Users
- Tags
- Questions
- Answers
- Bugs
- Library
- Package
- Class
- Functions
- Data (Class Variables)
- Interfaces
- Exceptions
- Transactions

Work started for Database Schema

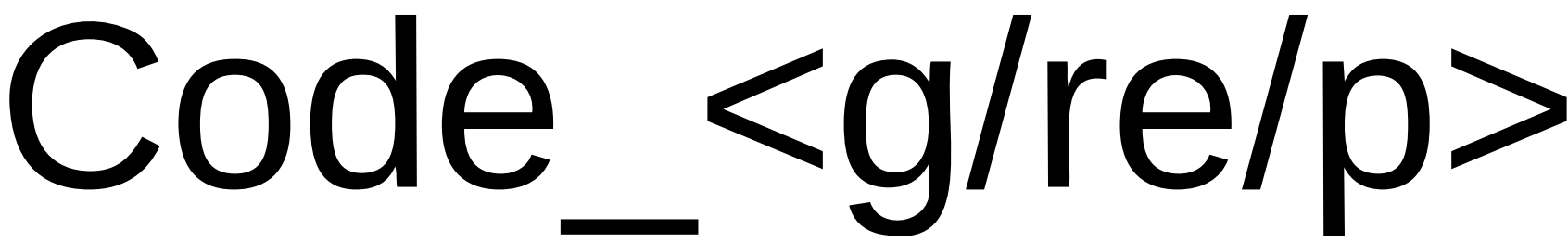
23rd January [Meeting 4]

Here are the role assigned to each member :

Roll number	Name	Role
2018007	Aditya	Implement the string parser and Bug Tracker page
2018069	Pranav	Q&A page and user page
2018100	Shikhar	Q&A page and user page
2018320	Tanupriya	Database for managing personal accounts, home page and bug tracker as user Interface.
2018357	Pragya	Databases for transactions and managing repositories as well as User Interface

Next we designed the database schema. Apart from data entities identified last week, we also identified some relations as entities that were frequent in queries. We drew the schema and implemented it.

Next Page for Schema



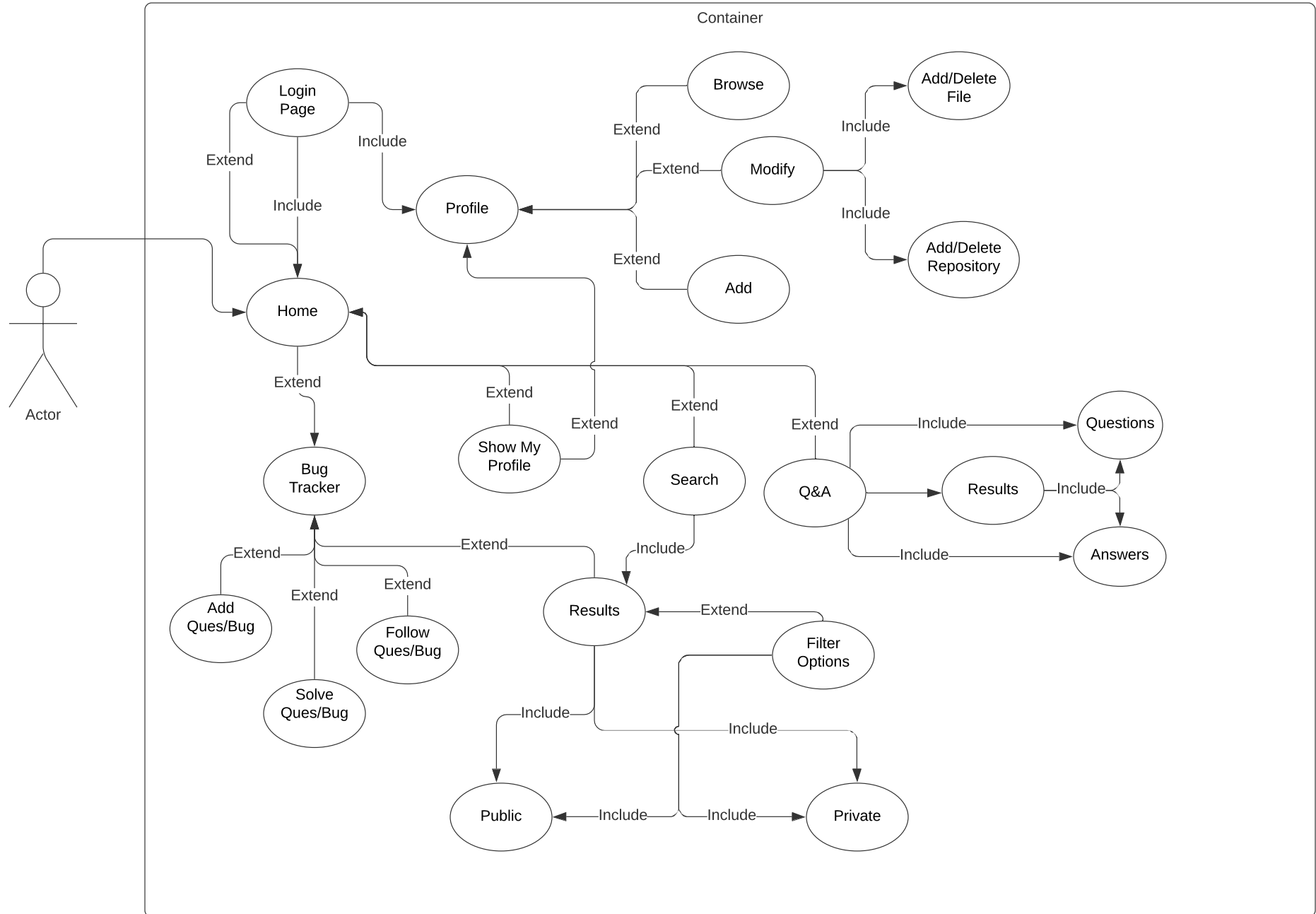
Week 4

30th January [Meeting 5]

We made the tables on sql using Data Grip. We had also set up constraints for each table. We also worked on the use case diagram for our application. We also brain stormed about the User Interface for our application. After making the tables in sql, we went ahead with populating the data. We gave 20 entries in each table. We divided the work equally among ourselves.

Here is the role each member performed after Week 4:

Roll number	Name	Role
2018007	Aditya	Designed the schema and Use Case. Worked on population of data
2018069	Pranav	Made the tables on Data Grip
2018100	Shikhar	Made the tables on Data Grip and populated the data
2018320	Tanupriya	Prepared the Use Case and populated the data
2018357	Pragya	Designed the schema and Use Case. Worked on population of data



Week 7:

We created indexes for our tables. Indexes were created by giving complete thought to the optimization.

The tables and their indexes are:

- answers ID,QUESTION_ID,ANSWERED_BY
- buginfile: FILEID
- bugs: ID
- classes: ID
- classhierarchy: CLASSID
- classimplementsinterface : CLASSID
- classinfile : FILEID
- downvoteroftbugs : BUGID
- downvotersofanswers
- downvotersofquestions : QUESTIONID
- downvotersofrepository : REPOSITORYID
- exceptions : ID
- exceptionthrownbyfunctions : FUNCTIONID
- files : ID,LOCATION
- filesinrepository : FILEID
- followerofanswer : ANSWERID
- followeroftbugs : BUGID
- followerofquestions : QUESTIONID
- followerofrepositories : REPOSITORYID
- functions : ID
- functionsinclasses : FUNCTIONID
- functionsininterfaces : FUNCTIONID
- interfaceinfile : ID
- interfaces : ID
- objectdefinitioninclass : CLASSID
- objects : ID
- parameters_of_function
- questions : ID,QUESTION, ASKED_BY
- repositories : ID
- tags : ID,NAME
- tagsofanswers
- tagsoftbugs
- tagsofclasses : CLASSID
- tagsofexceptions : EXCEPTIONID
- tagsoffiles : FILEID
- tagsoffunctions : FUNCTIONID

- tagsofinterfaces: INTERFACEID
- tagsofobjects : OBJECTID
- tagsofquestions : QUESTIONID
- tagsofrepositories : ANSWERID
- transaction : ID, REPOSITORYID,BUYER_ID,SELLER_ID
- upvotersofanswers : ANSWERID
- upvotersofbugs : QUESTIONID
- upvotersofquestions : QUESTIONID
- upvotersofrepository : REPOSITORYID
- users : ID, USER_ID

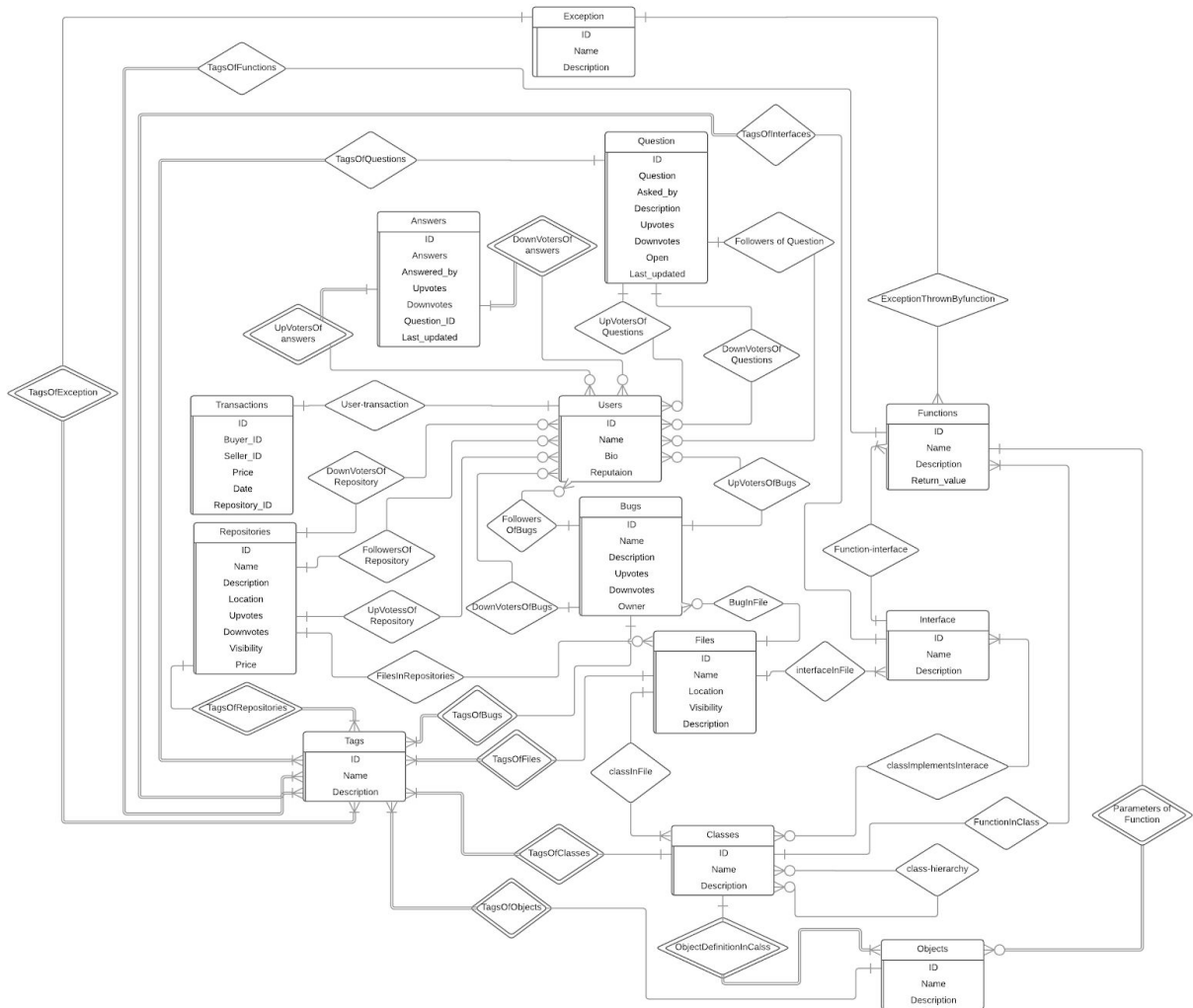
We also wrote the relational queries. They are listed below:

- $\sigma_{Name = 'Raymond'}(repositories)$
- $\sigma_{Upvotes > 7000 \text{ and } Downvotes < 3000}(questions)$
- $\pi_{Name}(\sigma_{users.ID = Questions.Asked_by}(\sigma_{Questions.Upvotes > 5000}(users) \times (questions)))$
- $\pi_{Name}(\sigma_{Price < 8000}(repositories)) - \pi_{Name}(\sigma_{Downvotes < 4000}(Students))$
- $\sigma_{Upvotes > 7000 \text{ and } Downvotes < 3000}(questions)$
- $\pi_{Name}(\sigma_{bugs.ID = tagsofbugs.BugID}(bugs))$
- $\pi_{Name}(\sigma_{Reputation > 7000}(users))$
- $\pi_{Name}(\sigma_{Visibility = 1}(files))$
- $\sigma_{Object_ID = 'object87'}(parameters_of_function)$
- $\pi_{ID}(users) - \pi_{Buyer_ID}(transaction)$
- $\pi_{Name}(\sigma_{bugs.ID = tagsofbugs.BugID}(bugs))$

Week 8:

The ER model is attached below

Code_Grep ER Diagram



The Strong entities identified are:

- Repositories
 - ID
 - Name
 - Description
 - Location
 - Upvotes
 - Price
 - Downvotes
 - Visibility
- Files
 - ID
 - Name
 - Location

- Visibility
 - Description
- Bugs
 - ID
 - Name
 - Description
 - Upvotes
 - Downvotes
 - Owner
- Question
 - ID
 - Question
 - Asked_by
 - Description
 - Upvotes
 - Downvotes
 - Open
 - Last_updated
- Users
 - ID
 - Name
 - Bio
 - Reputation
- Transactions
 - ID
 - Buyer_ID
 - Seller_ID
 - Price
 - Repository_ID
 - Date
- Interface
 - ID
 - Name
 - Description
- Exception
 - ID
 - Name
 - Description
- Classes
 - ID
 - Name
 - Description

- Functions
 - ID
 - Name
 - Description
 - Return_value

And the weak entities identified are:

- Tags
 - ID
 - Name
 - Description
- Answers
 - ID
 - Answers
 - Answered_by
 - Upvotes
 - Downvotes
 - Question_ID
 - Last_updated
- Objects
 - ID
 - Name
 - Description

Relations used are:

ParametersOfFunction
 Exception-function
 Function-interface
 User-Transaction
 User-Repository
 User-Question
 User-Question-Follow
 User-question-upvotes
 User-question-Downvotes
 User-Answer
 User-Answer-Follow
 User-Answer-Upvotes
 User-Answer-Downvotes
 User-Bugs
 User-Bugs-Follow
 User-Bugs-Upvotes
 User-Bugs-Downvotes
 User-Repository-Follow

- Tags of repositories
- Tags of file
- Tags of class
- Tags of exceptions
- Tags of interface
- Tags of functions
- Tags of object
- Tags of questions
- Tags of answers
- Tags of bugs
- Transactions of repositories
- Class-function
- Class-object
- Class-file
- Class hierarchy
- Class-interface
- Interface-file
- Bugs-File
- Repository-file

Week 10:

The key innovative feature. We planned on implementing tag hinting. It basically involves providing a hint to the user for a particular tag searched.

If given more time we could have implemented a string parser and file system to support our app.